---

1 September 2014

Hal Finney Messages on Cypherpunks 1992

Excerpted from http://cryptome.org/cpunks/cpunks-1992.zip

These messages alone do not convey the lively discussion of early public cryptography of which they are a part over five years, and continuing today, among thousands of participants. Finney continued to post to cypherpunks at least through 1997.

Cypherpunks mail list archive 1992-98: http://cryptome.org/cpunks/cpunks-92-98.zip (81MB)

```
From: Hal <74076.1041@CompuServe.COM>
Date: Sat, 10 Oct 92 11:04:58 PDT
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Mr. Squirrel?


Hi, I've just joined this list.

Interesting confusion about Mr. Squirrel.  That's one of the problems
with anonymity.  How do you know you're talking to the right person?

What you should do is to use a public key.  The pseudonym is not
really the name "Secret Squirrel"; anybody can use that.  The pseudonym
is the public key.  Any message signed by that particular key is from
that particular squirrel.  Any message you encrypt in Squirrel's public
key is readable only by him.  If Squirrel changes his key, he should
sign the message so you know it's really _that_ squirrel who's changing
his key (and not some other squirrel telling people to use a new key).

A pseudonym is a public key.

Hal

P.S. Here's my key, signed by PRZ:

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.01

mQCNAiqsNkwAAAEEAMKWM52m5EWi0ocK4u1cC2PPyHT6tavk9PC3TB5XBYDegf3d
sldRpnjJj1r+aO08FFO+QLEI9wtBqvf1PPP5iLX7sD2uIVlJH14MPtyVtjm9ZKb8
JMtCW74045BgtHBC9yQ3V7vXNV5jM6dE2ocnH4AI/pBFrGLJPKgTA69YIUw3AAUR
tCZIYWwgRmlubmV5IDw3NDA3Ni4xMDQxQGNvbXB1c2VydmUuY29tPokAlQIFECqu
M1Tidd4O/2f3CwEByrUD/3uoV2y+Fuicrrd2oDawgOw9Ejcx6E+Ty9PVPqKvflLs
0zYyGfeFVSgBbTSDP3X91N3F68nydl9J9VA6QRCGelHM1cZRukCJ0AYbKYfpwUN0
xjEGHsDrd2gT5iWlB3vBZvi+6Ybs4rSq+gyZzVm1/+oRrMen32fz2r0CLgUtHok2
=fF6Z
-----END PGP PUBLIC KEY BLOCK-----
```

```
From: Hal <74076.1041@CompuServe.COM>
Date: Tue, 13 Oct 92 08:55:18 PDT
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Re: Mr. Squirrel


||ugh Daniel raises some questions about using public keys to
verify pseudonyms:

>   Hal is somewhat right, anyone can use 'Secret Squirrel' and anyone
> can use any public key they want also.

But, once person A creates public key X, nobody else can sign messages
using X.  So if all messages from A are signed under X, we can know
that they are all from the same person, even if they are sent anonymously
```

or under a pseudonym.

> So, in a many-to-one scope (as
> in a maillist) where the sender can not use the one-on-one signed
> signiture method how do we have proff of who the sender really is?

You can use signatures even in a many-to-one scope. Messages from a particular person could be signed and the signature appended to the message. Then anyone who has the public key can check to see who the message came from. The process is a little unwieldy now in PGP because you have to separate the signature and message into separate files and run PGP on the signature file. This should be streamlined.

> [Good points about keeping track of key-pseudonym pairs]
> But all this needs to be done automaticly by the mailers and
> interfaces, else the system will be mis-used and folks will tire of
> the extra work that gets them little advantage.

Absolutely. The most crying need now, IMO, is to better integrate the cryptographic tools into mail readers and senders, so that it's not such a pain to use these things. People should be able to give a single command or press a button to decrypt an incoming message or encrypt an outgoing one. Only then will these features be used by average people.

There was a message posted on alt.security.pgp describing how to use PGP with the Emacs mail reading program. I'd like to see more messages telling how to use it with other systems.

Hal
74076.1041@compuserve.com

From: Hal <74076.1041@CompuServe.COM>
Date: Tue, 13 Oct 92 13:37:43 PDT
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: New remailer...

I have been experimenting with Eric's remailing software on the Sun 4 I use at work. This is what I've found.

First, Eric's descriptions of how all the different software components work together have been very helpful. The software has gone through three revisions as Eric added new features, so I implemented them in that order - first the basic remailer, then adding the "##" and "::" support for header management.

(I had to get perl and slocal before I could get started. Luckily my system already uses sendmail.)

Basically, I was able to put the parts together the way Eric described and have it work. I was able to send messages and have them remailed. I even did some tests bouncing mail between my remailer and Eric's.

Then I tried adding a new feature to the remailer - automatic message decryption using PGP. It's not really very secure since anyone with root privileges at my site can see my pass phrase, but my site is pretty isolated (a 2400 baud modem is the only link to the outside world). For this I had to add one line to Eric's model .maildelivery file to invoke my PGP filter, and had to write about a five line shell script to run PGP in a useful way. I am still tuning this a little bit but I can send the exact scripts out when people are ready for them.

One nice thing about this is that, with my remailer plus Eric's, and with the decryption option, you can now send anonymous messages for which no one person can tell that you did it. What you would do is to send the message first through Eric's remailer, so I don't know where it came from, then through my remailer, but with the message encrypted so that Eric can't tell where it's going after it leaves me. If more people will run remailers then we'll have much more security.

I will now tell you how to use it, in case you want to experiment. But remember that all messages are going across an intermittently-polled 2400 baud modem, so don't expect fast turnaround and please don't send a large volume of messages. Also, please don't pass

information about this remailer beyond this list, for now.

The remailer is at hal@ghs.com. The basic remailing operation is as Eric has described: either put "Request-Remailing-To: <dest>" in the header of the message, or put, as the first two lines of the body of your message:

::
Request-Remailing-To: <dest>

And follow these two lines with a blank line, then the message to be forwarded.

Decryption is just a little complicated. The thing to remember is that you want to do more than just have me decrypt the message. You want me to then remail the message after decryption. This means that you should prepare a message with remailing instructions as above, then encrypt the whole thing, including the "::" and "Request-Remailing-To:" lines. Encrypt using PGP with the public key I show below, and use the -a flag for Ascii output.

This will create a PGP output file, typically with the extension .asc. The first line will be:

-----BEGIN PGP MESSAGE-----

Now, you can send this message to me, but you have to do one more thing. You have to mark it as an encrypted message, by putting the line "Encrypted: PGP" in the header. If you can't put stuff into the headers of messages, then use Eric's "::" feature and add the following two lines, then a blank line, before "-----BEGIN PGP MESSAGE-----":

::
Encrypted: PGP

Don't forget the blank line after these two.

Now, this message can be sent to my remailer. It will be decrypted and then remailed to whomever you requested.

I know this sounds complicated, so let me break it down into steps:

1. Create the message.

2. Add "::" and "Request-Remailing-To: <dest>" and a blank line to the top.

3. Encrypt the whole file using PGP and the public key below.

4. Add "::" and "Encrypted: PGP" and a blank line to the top of the encrypted file.

5. Send it to hal@ghs.com.

That's not so bad, is it?

Now, if you're really adventurous, here's how to do the double-remailing process I described above, the one which keeps any one remailer from knowing who's talking to whom.

1. Create the message.

2. Add "::" and "Request-Remailing-To: <dest>" and a blank line to the top.

3. Encrypt the whole file using PGP and the public key below.

4. Add "::" and "Request-Remailing-To: hal@ghs.com", then a blank line, then "##" then "Encrypted: PGP", then a blank line, to the top of the encrypted file.

5. Send it to hughes@soda.berkeley.edu

The only complicated step is step 4, where you put in the remailing request to go from Eric's system to mine, and use the "##" line so that the outgoing message has "Encrypted: PGP" in the header.

If you want real security, encrypt the message using your friend's public key after step 1 and send that. Then nobody will even know what you're saying, let alone who you're talking to.

As promised, here's the public key for my remailer:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.01

mQBNAirY9EoAAAEB/iuDBqpeJ8gsNQwJNRYWBxH7uP95ApQ92CDhCmuSEJ0Tta0l
oCrC+8Br+D7Nfotb7hJlI0A1CYGAlmCsRO8VEmkABRO0H1JlbWFpbGluZyBTZXJ2
aWNlIDxoYWxAZ2hzLmNvbT6JAJUCBRAq2ISQqBMDr1ghTDcBARYlBADCjkCkIDvA
7QFtpYUlYjz/2U+/oDuMZBDlmAw8BCg3sdJG7hnxPE4yVgKoH/ozsb23pbFTPB8H
WNEjqTqixNybOKSKH9T8iCaRDA8+bS6xPN4YlWKD/Wg2EiyuOjD3v/vWgiZXzMR5
hpe0CYVJ6bM++hptXu+JxqDReJIot5FFbQ==
=p8FS
-----END PGP PUBLIC KEY BLOCK-----
```

Hal
74076.1041@compuserve.com

P.S. Coming soon: anonymous return addresses!

From: Hal <74076.1041@CompuServe.COM>
Date: Wed, 14 Oct 92 08:27:57 PDT
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Game items...

I'm trying to think in terms of things which were illegal but which have good moral connotations today.

Crosses and other Christian symbols were supposedly outlawed during the Roman empire (leading to the adoption of the fish as a symbol of Christianity). Posing as early Christians smuggling crosses ought to make the right-wingers happy!

Abolitionists had to smuggle runaway slaves out of the South on the so-called "underground railroad". Perhaps cryptography would have helped them coordinate their efforts.

Much of the support in the U.S. for freedom and privacy comes from our revolutionary heritage. I'm embarrassed at how little I can recall of what things were restricted in those pre-revolutionary days. I recall the Stamp Act and a few other laws, and I imagine that seditious materials were restricted. Perhaps the game players could be early revolutionaries trading items forbidden under British rule.

Hal Finney - 74076.1041@Compuserve.Com

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQCNAiqsNkwAAAEEAMKWM52m5EWi0ocK4u1cC2PPyHT6tavk9PC3TB5XBYDegf3d
sldRpnjJj1r+aO08FFO+QLEI9wtBqvf1PPP5iLX7sD2uIVlJH14MPtyVtjm9ZKb8
JMtCW74045BgtHBC9yQ3V7vXNV5jM6dE2ocnH4AI/pBFrGLJPKgTA69YIUw3AAUR
tCZIYWwgRmlubmV5IDw3NDA3Ni4xMDQxQGNvbXB1c2VydmUuY29tPokAlQIFECqu
M1Tidd4O/2f3CwEByrUD/3uoV2y+Fuicrrd2oDawgOw9Ejcx6E+Ty9PVPqKvflLs
0zYyGfeFVSgBbTSDP3X91N3F68nydl9J9VA6QRCGelHM1cZRukCJ0AYbKYfpwUN0
xjEGHsDrd2gT5iWlB3vBZvi+6Ybs4rSq+gyZzVm1/+oRrMen32fz2r0CLgUtHok2
=fF6Z
-----END PGP PUBLIC KEY BLOCK-----
```

From: Hal <74076.1041@CompuServe.COM>
Date: Sat, 24 Oct 92 09:02:41 PDT
To: <cypherpunks@toad.com>
Subject: Multiple messages + entropy

The Internet PEM (Privacy Enhanced Mail) standard uses the concept which Dean Tribble mentioned of multiple encryption (using each recipient's public key) of a single session key which encrypts the message. PGP's data structures do not currently provide for this but could be extended pretty easily to allow it.

On the entropy measure - I thought entropy was how many bits of information you get per character. Encrypted binary text would be pretty close to 8 bits per character. The RFC1113

Ascii encoding used by PGP reduces this to 6 bits per character
(e.g. a character set with 64 printable characters) neglecting
line separators and message beginnings and endings. So there
should be a little less than 6 bits per character for encrypted,
Ascii-encoded messages.

Hal
74076.1041@compuserve.com

From: Hal <74076.1041@CompuServe.COM>
Date: Sat, 24 Oct 92 21:47:19 PDT
To: <CYPHERPUNKS@TOAD.COM>
Subject: Re: remove from mailing list

What we _ought_ to do is to remind people that they should send
their unsubscribe notices to cyherpunks-request@toad.com, _not_
to the list address. That way we wouldn't all have to be bothered
by reading these messages, and the list volume would be that much
lower!

From: Hal <74076.1041@CompuServe.COM>
Date: Mon, 26 Oct 92 09:23:21 PPE
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Registering Keys...

This proposal to register keys was also mentioned in the July, 1992
Communications of the ACM, in an article by Ron Rivest, one of the
creators of the RSA algorithm. He was mostly criticizing the proposed
government Digital Signature Standard, stating that he thought that
the NSA was purposely trying to get "weak" cryptography installed as
the standard. Then he goes on to say,

"Are there technical alternatives that would satisfy all parties?
Perhaps. It is certainly the case that the formulation of the problem
to be solved has never been made explicit for the cryptographic
community to work on. I suspect that a solution based on 'escrowed
secret keys' might be workable, wherein each user is legally required
to depost his or her secret key with a trusted third party, such as
the user's bank. Cryptographic hardware and software would only
operate with public keys that were certified to having their corres-
ponding secret keys appropriately escrowed. A federal agency could
then obtain the secret key, or its use, with an appropriate warrant.
Once their secret keys were escrowed, multinational corporations could
even operate across borders with a high degree of authentication and
privacy (except perhaps from court-ordered wiretaps). Cryptographic
hardware and software manufactured in the U.S. would not operate
abroad without public keys suitably certified as having their secret
counterparts escrowed in the U.S. In an extension of this approach,
users can escrow their secret keys with several trusted third parties
in a 'secret-sharing' manner, so that no single third party can com-
promise the user's key. While this approach may have its own difficulties,
it does illustrate that weak cryptography is not the only technical
approach available. There may be much better techniques for achieving
a compromise between a number of conflicting national concerns."

At the time that I read this, I thought it was largely a rhetorical
device, making the point that if the government wants to infringe on
people's privacy, it should come out in the open and do so, rather
than skulking about. (Like saying, "if the government _really_ wants
to stop sexual immorality it would have to put a TV camera in every
bedroom".) And of course (I thought) this kind of proposal would never
be taken seriously. I'm shocked that Denning is now advocating it openly.

This proposal makes it illegal for people to communicate so secretly
that the government can't find out what they are saying. It could
apply to postal mail as well as email - it would be illegal to send
a message via post which the government couldn't interpret. If this
is really the government's purpose, then it should also require that all
private conversations be recorded, and the resulting tapes be "escrowed"
similarly in case the government needed to find out what was said,
for which it would have to get a court order.

As Tim suggested, this is probably a "trial balloon" being floated to

see what the reaction is. Let's see that it gets shot down.

Hal
74076.1041@compuserve.com

From: Hal <74076.1041@CompuServe.COM>
Date: Sat, 31 Oct 92 09:42:14 PPE
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Why I Don't Use PGP...

The best way to integrate PGP into other software is a tough question. There are so many different ways in which people read and send mail.

A lot of people receive their mail on some other machine, often a multi-user machine. So, the first question is, should the mail be crypted there, or should it be crypted on your personal machine. The second choice is preferable from the security point of view, but that means that you need to download at least your PGP mail in order to decrypt it, and it means you have to compose your mail on your home machine before encrypting, uploading, and sending.

(Phil Zimmermann had an idea for multi-user systems in which the RSA portion of the decryption, which involves your secret key, would be done on your personal machine, then the decrypted session key would be uploaded to the multi-user machine and the IDEA decryption done on the message itself. This way you only have to upload and download a couple hundred bytes regardless of the length of the message. This would require PGP to be integrated into a terminal program.)

If you _do_ download your mail before reading it, you could get in the habit of downloading _all_ of your mail into a big file, then running a word processor or some more specialized program to read the messages, one at a time, and compose replies. Then, when you were done, you could upload and send the replies.

If you worked in this mode, which probably few people do, you could integrate PGP by running it on the downloaded mail file before running your WP to read it. I have a Perl script which runs PGP on a file, finding the PGP messages in it, decrypting them, and replacing them _in_the_file_ with their decrypted versions. (Normally PGP outputs its decrypted contents to another file, which is a little inconvenient.) This can make PGP pretty transparent for decryption if you actually read your mail in this way.

Another possibility is to use a WP or mail reading program which has a "filter" mode, one which lets you pass incoming or outgoing mail through some program, and replace the mail with the results of that program. I don't know which programs can do this. A lot of Unix programs can, like VI and EMACS, but I don't know about PC's or other home machines. PGP has a filter mode which is designed to be used with WP's which can do this. There have been a couple of messages on alt.security.pgp which have advice on using PGP with various Unix mail reading programs. Mark Riordan's soon-to-be-released RIPEM program (an alternative, incompatible, RSA public-key program) has some ideas in its manual on how to use its filter mode with Unix mail, which mostly apply to PGP as well.

One other point: regarding a Mac port: There have been at least a couple of messages on alt.security.pgp over the past couple of months from people who have successfully compiled the PGP sources under Think C on the Mac. However, as a Unix/PC program, it ends up using a character window for I/O, which you type into just like you would on a PC. This is unacceptable for Macs, so nobody has released one of these. Still, compared to what Tim has to do, it would be an improvement. I think people should release their executables which work like this as an interim crutch version until the real Mac version is available.

Hal
74076.1041@compuserve.com

From: Hal <74076.1041@CompuServe.COM>
Date: Thu, 12 Nov 92 13:18:01 PST

To: cypherpunks@toad.com
Subject: No Subject

Bcc: Blind Recipients List:;
Subject: Why hardware random numbers?
Message-ID: <921112211037_74076.1041_DHJ46-1@CompuServe.COM>

I don't understand the desire for hardware-based random number generators.
It seems to me that a decent software RNG would be adequate for
the main uses that I have heard of for RNG's (mostly session key
generation).

Seed the RNG initially with a nice random set of characters typed in
by the user, plus timing information based on the rate of timing of
those characters. Also use the local system clock, and possibly a
hash of some sectors of the disk or some files on /tmp. Create a pool
of random numbers in this way.

As you use them, refill the pool, making the refilled bytes a function
of the current system clock, and whatever message you are encrypting
(or some other appropriate global state).

Use a nice strong RNG based on DES, MD5, IDEA, or some other cypher or
hash function.

I don't think anyone could break the resulting random stream without
a physical attack on your computer. Why pay $50 to $200 for a hardware
device when you can get the same effect in software that already exist?
Both PGP and RIPEM, I think, use the above techniques for their random
numbers.

Hal
74076.1041@compuserve.com

From: Hal <74076.1041@CompuServe.COM>
Date: Sun, 15 Nov 92 17:37:40 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Why remailers...

Tim mentioned that not many people on the list have expressed
interest in the remailers, and it occurs to me that maybe people
don't all share the vision of why this crypto technology is important.

I'm trying to recall how I learned about the possibilities of this
technology. I recall reading "True Names" a few years ago. Vinge had
his netters exchanging mail anonymously. The hero downloaded a big
batch of messages from a BBS and tried decrypting all of them to see
which were for him. Okay, I thought, that would be a way of disguising
which messages you were _receiving_. Then Vinge said something like
"and using more elaborate techniques, the sender of a message could
be hidden as well." Hold on, I thought. That will never work. If
they tap your line, they're going to know exactly what messages you're
sending. Too bad. Vinge had a clever idea going, but it's flawed.

I only learned about Chaum's crypto stuff last year. Somebody on the
Extropians list mentioned PGP, and I'd always had a casual interest
in crypto, so I downloaded it and played with it some. I thought it
was great and really got into it in a big way.

This got me interested in crypto in general, so I started doing some
library research. When I found Chaum's stuff, it just blew me away.
The first article I found, I think, was his CACM paper which is an
overview of many of the things that are possible. I started trying to
track down other papers by Chaum. Here were all the technologies
needed to make Vinge's world work, technologies which Vinge apparently
knew about long before I did.

It seemed so obvious to me. Here we are faced with the problems of
loss of privacy, creeping computerization, massive databases, more
centralization - and Chaum offers a completely different direction to
go in, one which puts power into the hands of individuals rather than
governments and corporations. The computer can be used as a tool to
liberate and protect people, rather than to control them. Unlike the
world of today, where people are more or less at the mercy of credit
agencies, large corporations, and governments, Chaum's approach balances
power between individuals and organizations. Both kinds of groups are

protected against fraud and mistreatment by the other.

Naturally, in today's society, with power allocated so disproportionately, such ideas are a threat to large organizations. Balancing power would mean a net loss of power for them. So no institution is going to pick up and champion Chaum's ideas. It's going to have to be a grass-roots activity, one in which individuals first learn of how much power they can have, and then demand it.

Where do the remailers fit in? They represent the "ground floor" of this house of ideas - the ability to exchange messages privately, without revealing our true identities. In this way we can engage in transactions, show credentials, and make deals, without government or corporate databases tracking our every move as they can today. Only by securing the ability to communicate privately and anonymously can we take the next steps towards a world in which we each have true ownership and control over information about our lives.

Chaum's ACM paper is titled, provocatively, "Security Without Identification - Transaction Systems to Make Big Brother Obsolete." The work we are doing here, broadly speaking, is dedicated to this goal of making Big Brother obsolete. It's important work. If things work out well, we may be able to look back and see that it was the most important work we have ever done.

Hal Finney
74076.1041@compuserve.com

From: Hal <74076.1041@CompuServe.COM>
Date: Sun, 15 Nov 92 23:37:38 PST
To: Cypherpunks <cypherpunks@toad.com>
Subject: Chaum's papers

A couple of people have asked for references to Chaum's papers.
The August, 1992 issue of Scientific American was mentioned here, I think.

The ACM paper I referred to is "Security Without Identification:
Transaction Systems to Make Big Brother Obsolete", October 1985.

The "DC-Net" is described in "The Dining Cryptographers Problem:
Unconditional Sender and Recipient Untraceability", Cryptology, 1988,
volume 1, p65-75.

The "Mix-net", which is similar to the remailers we are experimenting
with, is described in "Untraceable Electronic Mail, Return Addresses,
and Digital Pseudonyms", CACM, February, 1981.

Chaum also frequently presents papers at the Crypto conferences, so
if you can get access to the proceedings of these at the library you
will usually find one or two papers by him in each volume. However,
in recent years he has published on other topics which don't seem as
relevant to the freedom/privacy issues we are concerned with.

Hal
74076.1041@compuserve.com

From: Hal <74076.1041@CompuServe.COM>
Date: Wed, 25 Nov 92 09:17:58 PST
To: cypherpunks@toad.com
Subject: Remailers...

I think the remailing idea expressed via Tim (from David?) had
some nice features. It would be very easy to do replies to someone
whom you didn't know but from whom you'd received some anonymous mail.
As I understand it, if I send mail anonymously to David, he won't
(of course) know who sent it. If he replies, the mail will bounce
back to the forwarder. And the forwarder has remembered my forwarding
request so that it can send the reply back to me. After that it
deletes the remembered forwarding request for security.

I wouldn't object to this that much on security grounds; as David
pointed out, even a full implementation of Chaum's "mix" remailer
would fall to infiltration. Instead, I think there are some issues
involving usability.

For one thing, it sounds like this system is use-once as far as the anonymous return address. If David replied to me, then thought of something he wanted to add, his second message wouldn't get through to me.

Another problem is, what if two people send anonymous mail to David via the same forwarder. Then, when he replies, how does the forwarder know which of the two to forward the reply to?

It's also asymmetric, in that it will only work if one of the two communicants knows the true address of the other. A lot of the interesting features of Tim's "crypto anarchy" only arise if people can communicate without either one knowing the other's true address.

Let me mention a couple of other ideas which I've heard of for anonymous return addresses. One idea was posted several months ago on the Extropians list by Eric Messick. (Is he on this list?) It used a "pseudonym-based post office box". You would send a message to a remailing server saying, "Please save mail addressed to pseudonym XYZ123. I will pick it up later. Here is the public key I will use to authorize the pick-up, and here is some digital cash to pay for your trouble. Thank you." Then you send mail anonymously giving XYZ123@remailer.com as your return address. This mail stacks up at the remailer which saves it for you. At some later date you send a signed message to the remailer saying, "OK, send XYZ123's mail to me@me.com."

Eric Hughes had an idea which was somewhat like this but without the delay aspect. You would just set up an account with a remailer whereby all mail to XYZ123 would be forwarded to yourself. Then XYZ123@remailer.com would be your return address. This could include David's idea if you asked the server to delete your pseudonym after using it once.

All of these anonymous return address proposals can be enhanced by using a cascade or chain of remailers for your A.R.A.

Chaum's "mix" remailer would save up a batch of cryptographically protected messages, decrypt them, rearrange their order randomly, then send them out. This way if the remailer itself is secure but the network connections to it are being monitored, the correspondance between incoming and outgoing messages is lost. The other ARA suggestions could also benefit from this enhancement.

Chaum's idea for an anonymous return address was a somewhat more complicated form of the ARA I've implemented for my remailer. My ARA is simply a forwarding instruction, encrypted with the public key of the remailer. The advantage of this system is that you don't have to "register" with the remailer(s) in advance. It's less convenient than the other proposals, though, and there is the danger that the public key(s) of the remailer(s) involved would be revealed at some time in the future, which would then reveal that that old ARA really was you.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Wed, 25 Nov 92 13:04:57 PST
To: cypherpunks@toad.com
Subject: Electronic Banking

Some time back Tim May suggested that we should do some experiments with electronic cash. He offered to do some Xeroxing if people would "pay" him.

There are lots of proposals for electronic cash in the literature, mostly very complex. I think one of Chaum's simpler proposals would be adequate for email "banking". This proposal, from the beginning of his paper "Untraceable Electronic Cash" in Crypto 88(?), goes like this:

1. Alice chooses a random x and r, and supplies the bank with B=r^3*f(x) mod n, where f is a one-way function (like MD5), and n is the modulus for the bank's public key.

2. The bank takes the third root of B (e.g. via an RSA decryption) and sends it back to Alice: D = r * f(x)^(1/3), and withdraws one dollar from her account.

3. Alice extracts C = f(x)^(1/3) by dividing D by r. (Note that division can be done mod n without knowing the factors of n, but it's rather complicated.)

4. To pay Bob one dollar, Alice gives him (x, C).

5. Bob can verify that C = f(x)^(1/3), but he still has to send (x, C) to the bank in order to make sure that x hasn't been used before. Otherwise Alice could spend (x, C) twice. The bank increases Bob's account by one dollar.

This scheme is pretty simple and provides untraceability - the bank saw B and D but not C, so although it can verify that (x, C) is legit, it can't correlate that with Alice's withdrawal.

The main disadvantage of this approach is that Bob has to send (x, C) to the bank right away (or at least before sending Alice anything in return for her cash) to verify that the cash hasn't been used before. But in email, where turnarounds of a day or more aren't unusual, this should be tolerable.

Alice and Bob could be pseudonyms, using anonymous addresses to communicate with each other and with the bank.

Different denominations of cash could correspond to different exponents than "3" in the example above. (That is, $1 would use C=f(x)^(1/3), $2 would use C=f(x)^(1/5), $4 would use C=f(x)^(1/7), and so on.)

Technically, this would be quite easy to implement, using the code in PGP for the arithmetic, and MD5 for the one-way function. We'd need to define a few message formats. The RFC1113 ascii encoding from PGP could be used as well.

The "social" problems are more challenging, it seems to me. What is the backing for this electronic money? Why do people care what their bank balances are? Is this stuff really worth anything?

One possibility is to base digital cash on real money. People would open a pseudonymous account via email, then postal-mail dollars to the bank, enclosing their account number so the bank would know whom to credit with the deposit. Later, if someone wanted to withdraw "real money" from their account they would have to give a real postal address where it could be mailed. Now the electronic money is worth real dollars. Even if people didn't deposit or withdraw very often, it still has value because of the backing.

Unfortunately, this approach would currently be illegal (at least, unless you actually were a real bank!). If there were some way the bank itself could be anonymous, it might survive, but I don't see how to mail it money while keeping the anonymity. Still, we could consider experimenting with this on a small scale with accounts of no more than a few dollars. As long as it was clearly an experiment I doubt that any prosecutions would result even if it attracted government attention, because the expense involved in court costs would be so disproportionate to the few dollars involved in this technically illegal act.

Another approach would be not to try backing the digital cash at all, or rather backing it implicitly by the determination of various people to accept it and perform services or supply goods in return for it. Tim's offer to Xerox papers in return for digital cash would be one example. Perhaps others could provide some other services. It would be great if some shareware author would accept digital cash as a symbol of support for crypto anonymity.

One problem that I see with this approach is how you determine the size of the money supply. Or, in other words, how does new digital cash get started circulating? How do people get new accounts, and how much money is in them?

If these problems can be solved, a big advantage of this approach is that the banker can be anonymous. He would be known only by his anonymous address and his public key(s). This would provide some safety in the event that even a small-scale experiment like this was targetted for a crackdown.

Another issue is the prospect of multiple "banks", each issuing their own (incompatible) cash. How would they compete? Perhaps in terms of rapid turnaround? Some might choose to be anonymous, others would go public. The latter would have the advantage that people might trust them more, but OTOH there is more chance of your bank account disappearing after a crackdown for a public bank than an anonymous one.

Lots to think about here!

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Wed, 25 Nov 92 16:07:41 PST
To: cypherpunks@toad.com
Subject: Re: An easy-to-reply anonymous mail scheme

The problem with the idea of posting anonymous mail to a newsgroup is sheer volume. Remember, we aim at a system where a large fraction of mail is potentially being done this way. Imagine if almost all email was done today by posting to newsgroups! There is probably thousands of times as much email traffic as news traffic now. It would totally swamp the system. You'd literally have to send every email message sent by any user in the world to _every_ user in the world, in effect.

As Yanek says:

> You are guaranteed anonymity because no-one can find out who decrypted
> the alt.w.a.s.t.e message, since everyone received it.

This really won't scale to large numbers of users. Yanek also writes:

> > Here is an example of how to use the cryptographic remailer at
> > <hal@alumni.caltech.edu> to implement an anonymous return address.
>
> > But the again do you trust hal@alumni.caltech.edu...
>
> With conventional remailer schemes such as this one, you are announcing
> your True Name (or at least your True Internet Mailbox) to someone you
> must trust. With my scheme (posted earlier today) you don't need to trust
> anybody except yourself (to not make a dumb mistake like including a
> signature).

This is why you would want to use a chain of remailers as your return address, what Chaum calls a "cascade". No single remailer sees the correspondance between your anonymous address and your real address. Only by breaking all of them can the bad guys find out who you are. Ideally, remailers would operate in a variety of countries with different laws, making it difficult to crack them all.

Remailers could be designed to periodically flush themselves, deleting old keys and/or pseudonym maps. This way anonymous addresses would have a limited lifetime if desired, and the attackers would have only a finite time window to break all the remailers involved. (Different keys/pseudonyms could have different lifetimes as needed.)

We could also imagine that there are lots of remailers - not just dozens, or hundreds, but millions of them. Maybe almost everyone runs a "cheap" remailer on the side, collecting a few cents in digital cash for each message they pass, enough to pay for their own messages.

Putting all this together, you could have an anonymous address which passes through, say, 10 remailers which might be any of the millions of remailers in the world. It could have a limited lifetime of only a few hours for some ultra-sensitive applications, with the remailers involved flushing their databases after that time. To break this, the enemy would have to sequentially break into machines all over the world, one after another, defeat any physical barriers (locks, men with guns), overcome tamper-resistance in the computers, break the encrypted files, and find out what the next step is in the address cascade, all in a couple of hours. This doesn't seem possible.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Wed, 25 Nov 92 16:25:25 PST
To: cypherpunks@toad.com
Subject: Unlabelled PGP messages

Peter Shipley points out that PGP messages are labelled with an identifier of the person they are sent to. This hurts the anonymity of the messages somewhat.

What PGP actually puts in the cleartext message header is the "Key ID" of the recipient. This is the low-order 64 bits of the RSA modulus "n" of his key. (PGP displays only the low-order 24 bits when it shows key ID's, but it keeps 64 bits internally.)

I understand that there was some discussion during the development of PGP 2.0 of having a mode where this wouldn't be done. One possibility would be to output a key ID of all zeros for a message which wanted to hide who it was for. Then, as Peter suggests, it would either be trial-decrypted by all of the secret keys you have, or, more simply, it would just try your "default" secret key. Most people only have one secret key so both methods would be the same in most cases. Doing it the second way would be a pretty easy patch to PGP.

I'm not so sure now that this feature is that helpful. First, you don't have to put your real name in the "user name" field. You could use a pseudonym. So you really don't have to give much information away about yourself the way it is now.

Also, if someone is sending a message to you using encrypting remailers, they would encrypt it using your key, add remailing instructions for the last remailer in the chain, encrypt that using that remailer's keys, add remailing instructions for the next-to-last remailer, encrypt it again, and so on. (This would be done automatically by some future software - you wouldn't want to do this by hand!) The result is that the mail you send does not expose the key ID of your recipient. That information is only revealed when it comes out of the last remailer in the chain. And by that time, it's no secret, since that last remailer is using the true email address of the recipient anyway. So it's not giving anything away.

For the other kind of anonymous messaging, where you just post to a newsgroup or use some other kind of "broadcast" system, the key ID is revealed and for this case it might be better to hide it. But, the key ID can be useful in this application by letting you know which messages you should decrypt. No one has to know that a particular key ID is "you". You can still download 1000 messages and only read yours without anyone knowing which ones you read. But with key ID hidden you would have to decrypt them all to see which are yours. Do you want to decrypt all 1000? This will take minutes, hours, or days, depending on your key size and computer speed. (Most of the decrypt time is spent doing the RSA step, at least for most messages, and you can't tell if it's for you without doing that step.)

This still might be a good idea, but I'm not sure...

Hal Finney
74076.1041@compuserve.com

From: Hal <74076.1041@CompuServe.COM>
Date: Sat, 28 Nov 92 10:23:07 PST
To: cypherpunks@toad.com
Subject: Misc. Items

A few random points related to messages from the last few days.

(First, a "meta" point - whenever I post to this list, I get from 3 to about 10 messages over 2 or 3 days reporting on delivery errors. It would be nicer if these went to someone else. Some of the messages include as many as 20 or 30 names of list subscribers who were apparently included in the same "outgoing batch" as the bounced mail.)

On PGP key verification: I understand that Branko hopes to get version 2.1 of PGP out in a week or so. One of the new features will be a

mode to display a MD5 hash of each PGP key to facilitate read-aloud over the telephone. This should make it easier to phone-verify PGP keys, so we can have more _good_ sigs.

On pseudonyms and reputations: Several people have suggested that it would be easy to conjure up dozens of fake personalities who would then vouch for each other, giving the illusion of a well-founded and trusted pseudonym. This would be ideal for con men and cops.

This can be defeated by the use of the is-a-person credential, which Chaum describes in a couple of his papers, including CACM Oct 1985. This is a signed document given by an organization which makes you come in and give your thumbprint. The document is "re-blinded" a la Chaums' proposals for electronic cash, so that there can be no linkage between your is-a-person document and your actual thumbprint. However, the thumbprint makes it so you can't get more than one is-a-person document.

Now, when you go to apply for credit, and you say, here are signatures from dozens of people that I've done business with in the past, and I've paid them all off on time, the first thing the creditor is going to ask is, who are all these people? Are they legit? Can you at least show me is-a-person creds on them? You won't be able to. You only have one is-a-person credential, and you can't make more.

Again, these credentials do _not_ hurt crypto anonymity. There is no linkage between your credential and anything else about you.

On electronic banking: The interesting thing about electronic banking is digital cash. The key feature of digital cash is anonymity of payments. There is nothing subversive about this. Ordinary cash has (nearly) this property. Are you being subversive when you buy a newspaper without paying by check or credit card? Of course not.

The point is, we want to use digital payments so that we can transact business over the net. But the more things get computerized, the more possible forms of monitoring there are, by businesses as well as gov-ernments. There's nothing immoral in trying to keep VISA from knowing whom I like to do business with. Digital cash is designed to allow the convenience of electronic shopping, while keeping the privacy of ordinary cash payments. Conceptually, it's a simple idea.

Technically, what has to be done to turn an electronic banking proposal such as Don Bellenger's into electronic cash is some way to make it so that withdrawals can't be paired up with deposits. You also need, of course, to prevent cheating such as spending the same piece of cash twice. It's not trivial to meet these requirements. The Chaum proposal I described is the simplest one that I know of that achieves this.

On remailers: I haven't yet succeeded in doing a doubly-encrypted remailer test using Bill O'Hanlon's and mine. Once this works, I'll post instructions on how to do this, and possibly a script or two to make it easier. With two encrypted anonymous remailers, you can for the first time send anonymous messages such that no one person can know whom you are sending to. Bill and I would have to collude to find out who sent a particular anonymous message. If more such remailers can start operating, such collusion will become that much more difficult.

On John Draper: I just wanted to say publically that the famous "Captain Crunch" was an inspiration to me when I was in college in the 1970's. Although I did not become a "phone phreak" or "cracker" he represented to me the spirit of questioning authority and exploring beyond the accepted bounds of the system. I have followed his career to some extent over the years and I think he has more than paid for any sins he may have committed in his youth. I for one am thrilled to have the idol of my younger days on the list.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Mon, 30 Nov 92 11:19:02 PST
To: cypherpunks@toad.com
Subject: Re: Electronic Banking

Eric Hughes points out that the blind signatures used to provide anonymity in Chaum's digital cash schemes are patented (by Chaum

himself). This is a problem for an official, legal, profit-making business which wants to engage in electronic banking.

However, such a business would face many other problems. Chaum's digital cash system could be construed as using the RSA algorithms, since it is in effect an RSA signature which makes the cash unforgeable. So a license for RSA would have to be obtained as well. (RSA licensing would also be needed for secure communication between the bank and its customers.)

In addition, the acceptable use policies of NSFNET, which would probably have to be involved in most communications with the bank, are inconsistent with this kind of commercial activity, from what I understand. I believe new policies are in the works to allow commercial activities on the net, but these again will involve licensing costs.

There is also the question of the legality of private cash in any form, electronic or otherwise. Nobody seems to have hard evidence on this. On the one hand, people can legally exchange certain types of securities, and they could perform services for each other in return for such exchanges. This is legal, but they are supposed to report the income on their tax returns. Our digital cash would seem to fit this model. (But are such securities transfers as untraceable as digital cash? Perhaps not.)

On the other hand, I read several years ago that casino chips in Las Vegas were starting to be used as cash substitutes, but that the government cracked down on this practice. Perhaps this was too conducive to money laundering, especially with the reputed underworld activity in that city.

I do think, though, that an informal digital cash system, presented as a research project or an educational game, would be able to slip between the cracks of the legal system, much as PGP has done. And I think this presentation would be legitimate. Digital cash is new (actually, nonexistant, as far as I know), and any use of it would by its very nature be research and be educational.

I would suggest that anyone who proposes to implement such a game might want to consider releasing it anonymously (actually, pseudonymously). Sign the release with a PGP or RIPEM key, and let that be your pseudonym. Let people post messages to alt.security.pgp or some other newsgroup to discuss it, so that you can read them without revealing yourself, as proposed by Yanek. Post your replies anonymously using our remailers.

This way there won't be a single-point target for anyone wishing to punish users of the game.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Mon, 30 Nov 92 19:50:44 PST
To: cypherpunks@toad.com
Subject: Re: remailers

Fen Labalme quotes some messages with concerns about people needing permission from the sysadmins to run remailers.

The current remailers, based on Eric Hughes' approach, don't have this problem. I don't even know who the sysadmin is on the system where I run the remailer. They don't know anything about it. Eric's remailers don't require continual processes to be run, root privileges, hacking the mail tables, or anything special.

All you need is to be on a Unix system which supports the ".forward" file. This is typically implemented by the mail programs which deliver mail to the user's mail file. The programs check to see if a file called .forward exists in the user's home directory, and if so they treat its contents as either a program to pipe the incoming mail into, or a user name or list of user names to send the mail to.

This is the hook by which Eric's remailers operate. The .forward file is set up so that mail is piped to a program which sorts the mail based on its headers, using the mh utility slocal or, in my case, a

perl script which provides some of the same functions. Each message is checked like this, and if it doesn't contain any of the special stuff which activates the remailer software, it is simply deposted in the user's mailbox file as usual. Otherwise the remailers run and forward it as requested.

With this solution, there's no need for anybody to be aware that you are running a remailer, as long as it's not too much of a load in terms of extra message traffic.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Thu, 3 Dec 92 18:36:43 PST
To: cypherpunks@toad.com
Subject: Re: digital banking

I enjoyed seeing Karl's walk-through of Chaum's digital cash algorithm. The numbers looked right. One point is that doing different denominations isn't that much harder, you just need to have more exponents. As you generate your primes p and q, make sure that p-1 and q-1 aren't divisible by any small primes (other than 2). This will ensure that phi = (p-1)(q-1) is not divisible by small primes, hence that gcd(e, phi) will be 1 for those same small primes, in fact for all the odd numbers.

Karl also quoted a comment from Ray Cromwell expressing concern over proving deposits. Ideally you'd get a signed receipt from the bank for every deposit you make. In the case of electronic deposits, there exist protocols for "gradual secret exchange" that might be suitable for this purpose. What you'd like is to send the bank your deposit "gradually", while the bank simultaneously gradually sends you a digitally signed receipt. I don't recall the details of these protocols. Gradual exchange would not be very convenient for email because of the long turnaround times in mail systems, but if you had a better connection it might be useful, especially for large deposits.

Another way to look at it is, what stops the local merchant from taking your payment, putting it in his pocket, and then denying that you've given him anything? It would just be his word against yours. One answer is, if he does that to multiple people, their multiple stories would have more credibility than his denials. Similarly, a bank which made a habit of cheating its customers would find itself publically exposed. So it may be reasonable to trust the bank for routine transactions.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Fri, 4 Dec 92 09:55:49 PST
To: cypherpunks@toad.com
Subject: PGP questions

It's probably good to discuss how PGP works here, for educational purposes, but I would expect people to get the source code if they really are interested. I can give some pointers here to answers to some of the questions people have asked.

Steve Witham asks several questions. I think the crypto glossary which Tim posted a couple of weeks ago tells how RSA works, so I won't reiterate that here. Steve, if you didn't get it, maybe Tim could send it to you.

PGP's signature algorithm creates an MD5 message digest of the message, then signs that digest by raising it to the secret exponent "d", mod "n". MD5 is a public-domain message digest algorithm created by RSA Data Security, Inc, which breaks messages into blocks of 64 bytes as input and produces a 16-byte (128-bit) digest. PGP then pads this 16-byte number to be about the size of n, and does the exponentiation.

PGP does not do its decryption/signature exponentiation by actually raising the number to the power of d mod n, but by doing a mathematically equivalent operation involving two exponentiations, one mod p and one mod q. Since p and q are half the length of n, and exponentiation takes roughly the third power of the number of bits in the modulus, this reduces the amount of time by a factor of 4.

The random number generator has several different modes, and I can only suggest looking at random.c.

The data-compression algorithm is not really relevant but is based on zip. The binary-ascii translator is also not important but is a variant on the PEM standard.

Steve asks a lot of questions about the speed of different versions. Maybe asking on alt.security.pgp would produce some representative values. I'm not sure whether anyone has a database with all these numbers.

I understand that PGP 2.1 may have a faster version of the Upton modmult. Preliminary timings suggest that it's still slightly slower than the Merritt modmult on the Sparc, but if Merritt really is unreliable (I haven't heard about this) then switching to Upton will not involve much of a penalty. PGP is very fast on Sparcs anyway and I don't think making it 10% slower would matter.

Profiling indicates that yes, during the RSA encryption phase, most of the time is spent in modmult. An interpreter could be built to do a modular exponentiation using a C implementation of modmult and it would be about as fast. However, that still leaves the generation of the random session key, padding it with random numbers so that it is suitable for RSA operations, converting the RSA-encrypted session key into a format suitable for writing to a file, doing the IDEA encryption, wrapping this all up with control information so that it can be decrypted, looking up key information from a key ring file or some other data file (possibly decrypting it on the fly), converting to ASCII, etc. All these are part of an encryption/decryption cycle and can't really be skipped.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Fri, 4 Dec 92 10:03:39 PST
To: cypherpunks@toad.com
Subject: Re: Weakness of the PGP scheme ?

Vanguard@gribb.hsr.no asks about trying all the IDEA keys. If you will look in idea.h you will see that the IDEA key is 16 bytes long, which is 128 bits. This is long enough to make trying them all impossible. Trying to predict one IDEA key by knowing the previous one also looks hard, as PGP basically cycles IDEA on random input and takes the output as the keys. If you could predict this output it would be similar to breaking IDEA.

On the other hand, PGP normally keeps its random information in a small file called randseed.bin. It uses the contents of this file plus the current time of day in seconds as the input to generate the IDEA key. If you stole this file from someone (it's not cryptographically protected, unlike the secret key ring), and you know within several hours or a day when he sent each message, you could probably calculate all possible IDEA keys in a feasible amount of time (by trying all plausible values for the time of day in seconds). This would also let you calculate the new contents of the randseed.bin file. As long as you didn't miss any messages he sent, you could keep doing this and break all of his outgoing messages.

You can prevent this by removing your randseed.bin file and substituting an empty file (or one that is less than 16 bytes long) in its place. This will cause PGP to prompt you for random keyboard input each time you send a message, which would make it impossible for the attack above to work. It would mean less convenience, though.

The relevant routines are make_random_ideakey() and strong_pseudorandom() in crypto.c, as well as the code in random.c.

Hal

74076.1041@compuserv.ecom

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Sat, 5 Dec 92 17:25:29 PST
To: cypherpunks@toad.com
Subject: Anonymous address problems

We've had some discussion of anonymous addressing, which allows someone to post an address at which mail can be sent to them without people being able to find out exactly who they are.

I showed how the current remailers could, somewhat clumsily, allow anonymous addresses. The problem is, with a single-remailer anonymous address that remailer sees whom each anonymous address corresponds to, so you have to trust it. Now that other encrypting remailers are up it's possible to have anonymous addresses which go through more than one remailer before going to the final destination. This, I thought, would provide a more secure address since a whole group of remailers would have to be "broken" in order for someone to find out where a given anonymous address leads.

However, with the current implementation, there is a security weakness. Whomever owns the last remailer in the chain for your anonymous address can find out who you are. They do this simply by sending an anonymous message with known contents, like "test number 1598293". They then watch all messages going through their remailer, looking for one whose contents match what they sent. If they are the last remailer in the chain, when they see this message go through them, they can look at whom it is being sent to, and so they then know your true name. So, a multi-remailer anonymous address is really no more secure than a single-remailer one.

Chaum, in his "mix" paper, avoided this problem by having the anonymous addresses include a random number which each remailer sees as it decrypts the incoming message. (There is always such a number, it turns out, for the RSA encryption to be secure.) He had the mix, as it passes the message through, encrypt the contents with a single-key algorithm (like DES) using this random number as the key. This way the message is transformed at each step and so if it later comes back through the same mix, it won't be recognizeable as the one it sent earlier. So the attack above fails.

For this to work, the user has to save the random numbers that were used to construct his anonymous address, and decrypt the message using DES with these as keys before going on to read it or public-key decrypt it as usual. This would be quite a bit less convenient.

Chaum goes on to say that these return addresses can only be used once. I was a little puzzled by the exact attack that he is trying to defend against in applying this rule. Chaum doesn't always make the attacks clear, leaving that as an exercise for the reader. I believe the problem is that, say, the last remailer in the chain could send 100 messages to a given anonymous address (all would have different contents). Then, after working its way through the remailer chain, it would see 100 messages going to the same final destination. It could guess that those 100 were the 100 it sent, especially if it repeats this test every few days with different numbers.

Chaum's rule of allowing each anonymous address to be used only once makes them much less useful.

These complications just go to show that real security doesn't come easy or cheap. There is still work to be done before we achieve the goal of crypto anonymity.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Mon, 7 Dec 92 19:51:29 PST
To: cypherpunks@toad.com
Subject: Anonymous address problems, etc.

Chris Hibbert sent me a solution for the problem I mentioned about the last remailer in an anonymous address chain being able to discover the true name for the address. He suggested making your own remailer be the last one in the chain. If you added a little padding, so the second-to-last remailer couldn't tell (from the small size of the encrypted address) that yours was last, this sounds pretty good. And it's another reason to run a remailer, even if you had to do it manually.

On the digital cash issue - I think we should get some kind of implementation of digicash (oops, "electronic money") out for people on the list to play with. Then, we should think of some kind of experimental email-based game to play which would use the special characteristics of anonymous cash. Maybe we could use the remailers, too. Players would send cash back and forth to each other as part of the game. Even if the tools are sort of rough at first, this could show where the most work is needed.

Can anyone think of a good kind of game that we could play? As Eric Hughes pointed out, the cash doesn't have to be "cash", it could represent "gold" or "carrots" or any other material which exists in limited quantity.

I think it was on the extropians list that Eli Brandt pointed out the need for a shorter term than "digital cash" for this cryptographic money. He suggested "cryps". I thought of "emoney" or "ecash" by analogy to "email". Here's another one: "crydets", based on "credits". Or maybe we should call them "Chaums". That way he'd be less likely to sue us for infringing on his patent. ;-) I notice he managed to cleverly name DC-nets after his initials so maybe he'd like this.

Hal
74076.1041@compuserve.com

P.S. I hear PGP 2.1 is coming out today.

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Tue, 8 Dec 92 14:53:49 PST
To: cypherpunks@toad.com
Subject: PGP key generation

Steve Witham asks about PGP's "peasant_modmult" routine, how it can subtract the modulus on each addition. It doesn't, but the code is a little misleading (taken from mpilib.c, PGP 2.1):

```
    while (bits--)
    {   mp_shift_left(prod);
        msub(prod,pmodulus);    /* turns mult into modmult */
        if (sniff_bit(multiplier,bitmask))
        {   mp_add(prod,multiplicand);
            msub(prod,pmodulus);        /* turns mult into modmult */
        }
        bump_bitsniffer(multiplier,bitmask);
    }
```

What is confusing is that "msub" looks like "mp_sub". Actually, msub is a macro defined in mpilib.h:

```
#define msub(r,m) if (mp_compare(r,m) >= 0) mp_sub(r,m)
        /* Prevents r from getting bigger than modulus m */
```

So, msub actually only subtracts if it's necessary, as would be expected.

Jim McGrath asks:

> When PGP generates keys doesn't it always pick d and e to have
> the same number of bits, ie 446 for the strongest type?

d and e are the decryption and encryption exponents. e is chosen to be small, typically the number 17 or slightly larger. d is then about the size of n, your key modulus.

p and q, the two primes which are multiplied to get n, are about the

```
same size, usually.  PGP does have some code to make sure they aren't
too close to the same size (taken from rsagen.c, PGP 2.1):

        /*      See if p and q are far enough apart.  Is q-p big enough? */
        mp_move(u,q);   /* use u as scratchpad */
        mp_sub(u,p);    /* compute q-p */
        if (mp_tstminus(u))     /* p is bigger */
        {    mp_neg(u);
            too_close_together = (countbits(u) < (countbits(p)-7));
        }
        else            /* q is bigger */
            too_close_together = (countbits(u) < (countbits(q)-7));

        /* Keep trying q's until we get one far enough from p... */
    } while (too_close_together);

What this does is make sure that |p-q| is > 1/128 of the larger of p
or q.  This is designed to make sure that p and q are both very far
from the square root of n.  I.e. if n were 1024 bits, p and q could
both be about 512 bits, but they would be at least about 2^505 from
the square root of n, foiling the square root attacks.

Hal
74076.1041@compuserve.com
```

```
From: Hal <74076.1041@CompuServe.COM>
Date: Sun, 13 Dec 92 20:43:46 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: dcnets...


-----BEGIN PGP SIGNED MESSAGE-----

I thought I would reply here to Yanek's message about dc-nets
since it is a topic that should be of interest to the list.

I think it's great that someone is going to start experimenting
with these systems.  The sooner we start playing with this technology,
the better.

I have a few general comments about the system.  First, I like the
idea of splitting the key information management from the communication
management.  That way people can choose their level of security, all
the way up to one-time pads if they want.

However, I think there should be another system chosen for exchanging
key information initially than using PGP to send large one-time pads.
Dc-nets really eat up the bandwidth.  Yanek estimates using 125
kb per message sent!  And having to send one-time key information
around doubles the bandwidth.

Also, Eric Hughes pointed out to me in private mail once that a system
like this for key information exchange is only computationally secure.
You are basically relying on RSA and IDEA not to be broken.  As long
as you're doing that for this initial experiment, why not save yourself
a lot of trouble.  Just exchange an IDEA (or DES) seed, then cycle it
repeatedly through IDEA (or DES), taking the low order bit or few bits
as new random ones.  If two people have the same seed, they will generate
the same random bits.  And if IDEA is secure, your bits should be
secure.  If they aren't, PGP isn't secure.  PGP has code to do this.
I think it's in the IDEA.C module.  Also see strong_pseudorandom in
CRYPTO.C.

So, I'd suggest that the key exchange part just exchange a short key
and then a program generates the new random bits as needed for the
messaging.  Keep the key stuff separate, though, so people really can
do one-time pads if they want to eventually.

Another point is the amount of messaging people will do.  I think the
system should be enhanced to allow people to send and receive messages
to/from non-DC-net participants.  Otherwise you have 10 or 20 people
who hardly know each other.  What will they have to say to each other?
You won't get a good picture of message loads.

I don't foresee everybody in the world being hooked into interlocking
DC-nets any time soon (if ever).  But I do think there will be DC-nets
for some interested people.  They will achieve anonymity amongst
the group for messages sent beyond the group.  In other words, it will
```

be known that a message comes from a certain DC-net group, but it will be impossible to tell which person in the group sent it. Likewise, messages could be sent to a DC-net group without it being known to whom they are sent.

I think this capability should be added very soon to the DC-net software. It sounds like the software may include automatic message-sending capability and if so, something which just recognized a special message header and took it as "Request-Remailing-To:" should be easy to add. Likewise, incoming messages to the Dc-net (which would be sent to some random person in the group) should be easily forwardable to the DC-net system from outside. I don't have a clear picture of the user interface from Yanek's description so I'm not sure how easy/hard these would be to do.

One other thing I'd mention: the mechanism of reserving a slot and then sending a message is discussed at some length in the Ph.D. thesis of Jurgen Bos. Tim May kindly sent me an excerpt from the thesis which included this discussion; I think it may have been part of the original cypherpunk meeting handout. Bos compares several message reservation schemes and discusses which are the best. It might be good for Yanek to take a look at this document.

Yanek asks about sending encrypted traffic over amateur bands. This is definately illegal in the U.S. The reason is that the amateur bands can't be used for commercial purposes, so the FCC demands to be able to know everything that is being sent to make sure this rule is being complied with. However, there are some commercial packet-radio systems starting up and presumably they won't be subject to this limitation.

It may not be practical to incorporate all of these suggestions at first, but I do think that using PGP to exchange a RNG seed would be better than using it to exchange one-time pads. I'm looking forward to seeing the system in operation.

Hal
74076.1041@compuserve.com

-----BEGIN PGP SIGNATURE-----
Version: 2.1

iQCVAgUBKyvk8qgTA69YIUw3AQHyuQP/fXIkyCWR5GCiZsiRvMThcJK5xMbOQEIF
ow9S9xQ+7kiiJuF4dVp7NRyNTBjO2tBiQDh4JRKb4Pl7LGq+KKYQSTDzGgEo7hOw
dkgujwxbCAXjn2XEMewRHprZMPV4XB+iGIZzQ4piqubzWg8hOV8sMhduGaHKnhGc
MlhbbmhToOc=
=+cPN
-----END PGP SIGNATURE-----

Distribution:
CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM

From: Hal <74076.1041@CompuServe.COM>
Date: Mon, 14 Dec 92 10:31:00 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Remailers.

Tim's message brings up a point I've been wanting to mention. The prototype remailer software keeps log files of all messages passed through it. There are different reasons why people running the software might wish to have these logs. One purpose is for debugging; the remailers don't produce much in the way of error messages and the log files can be useful for tracking down errors.

A few weeks ago, for example, one user was having difficulty sending messages through my remailer, and he posted here about it. I was able to confirm that his messages had come in and been sent out.

However, another possible reason is for the case of abusive messages. I had one message go through that appeared to be directed towards the sender's boss, and was rather unfriendly in tone. The remailers give the outgoing messages the superficial appearance of having come from me. This message wasn't that bad, but there's nothing to stop someone from sending a really vicious, racially or sexually harrassing message, and I am very concerned that I could get in trouble for that.

What I've generally done is to delete the log files every few

days, usually after a quick perusal to see if there are any
messages which the recipient might object to. Sometimes if I see
a message which is of an illegal format so that it didn't get sent,
(like forgetting the ":" in "Request-Remailing-To:") I'll send a
message to the sender telling him what he did wrong.

I feel that people who run remailers should set their own policies
as far as the confidentiality of the messages they forward. Running
a remailer can be somewhat risky in the current climate and the
operators can legitimately seek whatever level of protection they
are comfortable with. However, I think it would be good if the users
of the remailers could get some information about what the privacy
policies are. Maybe some remailers will simply not keep logs; maybe
others will keep logs but not look at them unless a specific circumstance
arises, and so on. Eric Hollander has been creating a list of remailers;
perhaps he could solicit this kind of information from the operators
and publish it along with the remailer addresses and keys.

Hal
74076.1041@compuserve.com

Distribution:
CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM

From: Hal <74076.1041@CompuServe.COM>
Date: Thu, 17 Dec 92 22:53:06 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Positive reps.

-----BEGIN PGP SIGNED MESSAGE-----

I was thinking about the idea of positive reputations and thought I'd
mention a few thoughts while waiting to see Dean's notes.

As I understand it, a positive reputation is basically some kind of
recomendation or endorsement of a person by someone else. It might
be fairly specific, like "so-and-so paid a debt of $50 to me within
the agreed-upon two-week period." Or it could be more general, like
"so-and-so is, in my opinion, an honest person."

I was thinking of positive reputations that might be relevant to the
problem mentioned of anonymous posting to mailing lists and newsgroups.
As Dean mentions, we can envision a system which uses the opposite of
kill files. Instead, messages would only be displayed from people who
met certain criteria in terms of their reputation credentials.

What would I want to see in the way of such credentials that would help
me decide whether I wanted to read the message? (The issue of judging
the validity of credentials is discussed below.) Maybe recommendations
in favor of the poster's intelligence, knowledge, judgment, temperance
(i.e. reluctance to flame), etc. would be useful. Imagine a system in
which a person was rated from 1 to 10 in each of these categories. A person's
positive reputation would consist of (digitally) signed statements from
various "endorsers" (or "introducers"), each giving their numeric judgements
about the person in question.

With this system, I could set my ideal mail/news reader to only display
postings from people whose scores met some standards. Maybe I would
average them; maybe I would weight the different categories according
to my own tastes. But this would let me filter out time-wasters like
the random poster who was causing problems.

Now, this still leaves open the problem of judging the validity of
various credentials. This problem is very similar to the problem of
accepting key signatures in PGP. If I receive a PGP key loaded with
signatures, that doesn't mean much unless I know at least one of the
people involved (either directly, or through the net). Only then can
I judge how valid the signatures are.

In the same way, if a new person posts on a newsgroup, and includes his
credential loaded with 10/10/10/10's, that doesn't mean anything unless
I know some of these people who are vouching for him. In the most
extreme case, he could have created a bunch of false identities and had
them provide all the endorsements. So an endorser unknown to me is
not useful.

It's interesting that the PGP key "web of trust" may have application
to this more general problem. I wonder if the PEM folks will push for
a centralized "email posting quality" hierarchy in which agencies rate
each person's quality of postings and assign them an official score. :-)
This would be the analog of their solution to the key-signature problem.

That's about as far as I've gotten. A few more general comments:
Such a system requires an infrastructure of public-key encryption and
digital signatures. Only in that way will signed credentials be secure
enough to be meaningful. Since so few people use these tools today,
a positive-credential system would filter out almost everyone, throwing
out the baby with the bath water.

But, remailers are springing up everywhere. It is an idea whose time
has come, it seems. So the problem exists today, but the solution can't
be practically applied for at least another year or two, even assuming
rapid acceptance of PGP, RIPEM, PEM, or some other cryptographic standard.
That means that there may be some political pressure against remailers
during this interval. Perhaps we can turn this to our advantage by describing
the advantages of a credential system, and using that to further encourage
widespread use of cryptographic programs.

Hal Finney
74076.1041@compuserve.com

-----BEGIN PGP SIGNATURE-----
Version: 2.1

iQCVAgUBKzFI+KgTA69YIUw3AQEdCQP8DbPD9jrlR1MhlLOOQrSUc8Svcue2DZsj
+DIXiC50bpv+C5pZYtoCa5SuOXX8W6XmZRSkZW3gilvEKDQ2Zt7hH0ol+tFnn8cs
q05T1bYJIZqdMdqia6PZkVyvs+DQLuQSog5rZxAja1XfC/Rq59RnbPp2IViLqDiD
OfiasXA4Egs=
=GVH7
-----END PGP SIGNATURE-----

Distribution:
  CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM

From: Hal <74076.1041@CompuServe.COM>
Date: Sat, 19 Dec 92 16:52:59 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Remailers.

-----BEGIN PGP SIGNED MESSAGE-----

Responding to Edgar Swank's message:

> We've already discussed use of PGP by pedophiles to encrypt
> incriminating diaries. It gets better. PGP and remailers can
> be used to encrypt and ship "kiddy porn" GIF files. Even private
> possession of such material is a serious crime in the USA. A chain
> of anonymous and encrypted remailers could be used to drop kiddy porn
> into newsgroups like
>
>   alt.sex.pictures
>
> I'm sure that would create quite a stir!

This could easily happen soon, as news of these remailers spreads.
There's been talk on the net about a student at an Ivy League college
who is being investigated by the FBI for allegedly posting illegal
images containing child pornagraphy. Once people find out about
anonymous remailers, especially ours with their chaining capabilities
and encryption, they will realize that such posts can be made with
almost total safety. Presumably this may increase the number of
people who would attempt it.

> Perhaps Duncan or another attorney would like to comment on legal
> liability of remailers in these kinds of situations. I don't think
> there's currently any law on what records remailers would have to
> keep; if you didn't have them, you couldn't give them up.
>
> If this becomes a problem, expect legislation regulating or even
> outright banning remailers. However, if remailer code has spread
> over many national boundaries, such legislation may not be effective.

My guess is that it will not be feasible to ban remailers, since it would be hard to draw the line between completely automated remailers, and simple manual forwarding of a message that someone sent you, which happens all the time and can hardly be made illegal.

I suspect that instead the approach would be to claim that remailer operators are responsible for the material their remailers produce, regardless of its original source. So if child porn comes out, I am guilty of sending child porn. I can argue that my remailer was automatic and that I shouldn't be held responsible for what comes out of it, but my guess is that this argument will be rejected on the grounds of personal responsibility, and because no one forces me to run a remailer which sends out anything that comes in to it.

Such a policy would be a plausible extension of current Internet policies, IMO. RFC 822, the document which describes the format of Internet mail message, in session 4.4.2 discusses the "Sender:" field, and says, "Since the critical function served by the 'Sender' field is identification of the agent responsible for sending mail and since computer programs cannot be held accountable for their behavior, it is strongly recommended that when a computer program generates a message, the HUMAN who is responsible for that program be referenced as part of the 'Sender' field mailbox specification." [Capitalization in the original.] The need for a person to take responsibility for each piece of mail that is sent would tend to lead to the policy I mentioned.

With such a policy in place, if enforced by law, I don't think people would run remailers in this country because of the legal risks. There would still be the international remailers, though.

> By the way, no-one has yet responded to my request to add features
> to Eric's remailer to 1)remove the "sig" after "--" and 2)Anonymously
> post to newsgroups available at the remailer site. Both these
> features are present in the Australian remailer at Pax. Unfortunately
> that remailer seems less secure than Eric's in that it keeps more or
> less permanent records of its users at the remailer site and it
> doesn't seem to be able to chain to other remailers.
>
> --
> edgar@spectrx.saigon.com (Edgar W. Swank)
> SPECTROX SYSTEMS +1.408.252.1005 Silicon Valley, Ca

It should be possible to chain from Pax to our remailers, getting the best features of both. Pax could be the first remailer in the chain, stripping the sig. The message could then go through our remailers, providing non-traced security, at least if you can find someone who does not keep logs.

Anonymous posting can be done already. Anyone can post to most newsgroups by sending mail to certain addresses. One such system is at Berkeley. Send to, e.g., sci-crypt@ucbvax.berkeley.edu to post to sci.crypt. There is another such system running in Austin, TX, I believe, but I don't have the corresponding net address handy. So this capability is already provided by our (or anyone's) remailers.

One reason I haven't looked at stripping the .sig is due to an email discussion I had about a month ago with Eric Hughes. Eric strongly felt that message bodies should be preserved as much as possible by the remailers, in accordance with the general principle of Internet mail forwarding. Too much mangling is done already by mail gateways, and adding more changes might be harmful. Obviously, the remailers can't avoid doing some processing of the message body, what with the "::" pasting tokens and PGP decryption, but Eric felt that unnecessary changes should be minimized. I know Eric is working on some new remailer concepts so I want to defer to him on this issue for now.

Hal Finney
74076.1041@compuserve.com

-----BEGIN PGP SIGNATURE-----
Version: 2.1

iQCVAgUBKzOXf6gTA69YIUw3AQEvQQQArHXSwwrkQ3mSGOD60ZBV/p1R9RONqv2x
S64iJjdJB43G2nxLHB2t9xXAKSdCT00shEtil8LWu20XW0rWXqmMXyYudtt3qf3t
2arqlvxdGXlPt3eISFJ97U6jaI1EhswPg29fjuXMMaKv5OO/gBKp1i9Cig7suZDt
EkUDJihLQxc=
=q6zH
-----END PGP SIGNATURE-----

Distribution:
CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM

From: Hal <74076.1041@CompuServe.COM>
Date: Tue, 22 Dec 92 09:03:06 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Remailers.

-----BEGIN PGP SIGNED MESSAGE-----

One problem with the current encrypted remailers that people should
be aware of is that, since they operate unattended, they have to be
able to decrypt messages automatically.

This means that when an incoming messages arrives, the remailing software
automatically runs PGP on the incoming message to do the decryption.
But to decrypt, PGP has to be given the pass phrase for the remailer's
secret key. The only way this can be done is to have the pass phrase,
IN THE CLEAR, in the remailing software scripts.

The scripts are (or should be) protected using the Unix file system
so that only the owner and root can read them. But it's important to
know that root has access both to the secret key ring which holds the
remailer's key, and to the pass phrase which will activate that key.
This means that, at any time, root can find out the secret key of the
remailer, and read all messages encrypted for that remailer.

I don't think there is any way around this problem if the remailer is
going to run unattended. The only real solution is to operate on a
machine where it doesn't matter whether root knows the key; that is,
a machine where root is the operator of the mailing list.

Hal
74076.1041@compuserve.com

-----BEGIN PGP SIGNATURE-----
Version: 2.1

iQCVAgUBKzcZ0agTA69YIUw3AQEFhAQAlloC1eyaIuJDG91VzPoRv0MjzKlob8Te
C3N7XWLvszypOgKNBoEb1z8fF5ZsS20NhhRhtr7A3J4jxe88vGuea0Kvxzj4NGd4
bQeewhYk01ygoLzZOwv8BnN6pE7/uxk5POWq3XQuX80VQzistePfRRdNozTA09EY
4bBOr3s9Ig8=
=nJ/a
-----END PGP SIGNATURE-----

My public key, signed by PRZ:

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.0

mQCNAiqsNkwAAAEEAMKWM52m5EWi0ocK4u1cC2PPyHT6tavk9PC3TB5XBYDegf3d
sldRpnjJj1r+aO08FFO+QLEI9wtBqvf1PPP5iLX7sD2uIVlJH14MPtyVtjm9ZKb8
JMtCW74045BgtHBC9yQ3V7vXNV5jM6dE2ocnH4AI/pBFrGLJPKgTA69YIUw3AAUR
tCZIYWwgRmlubmV5IDw3NDA3Ni4xMDQxQGNvbXB1c2VydmUuY29tPokAlQIFECqu
M1Tidd4O/2f3CwEByrUD/3uoV2y+Fuicrrd2oDawgOw9Ejcx6E+Ty9PVPqKvflLs
0zYyGfeFVSgBbTSDP3X91N3F68nydl9J9VA6QRCGelHM1cZRukCJ0AYbKYfpwUN0
xjEGHsDrd2gT5iWlB3vBZvi+6Ybs4rSq+gyZzVm1/+oRrMen32fz2r0CLgUtHok2
=fF6Z
-----END PGP PUBLIC KEY BLOCK-----

Distribution:
CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM

From: Hal <74076.1041@CompuServe.COM>
Date: Tue, 22 Dec 92 09:02:47 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Remailer encryption.

-----BEGIN PGP SIGNED MESSAGE-----

From: miron@extropia.wimsey.com (Miron Cuperman)

> In any case, the current scheme for ARA's is insecure. This is
> because people can send plaintext messages attached to the ARA.
> This allows breaking anonymity by monitoring of the traffic from
> all remailers and waiting until the message appears at one of
> the outputs.
>
> I will implement a more secure scheme. The ARA will include
> encryption instructions for each remailer. Since each remailer
> will be doing a transformation on the message, the attack above
> will not be feasible.

I'd like to hear more about this plan. What kind of encryption instructions would be used in the ARA? Would they be public key or secret key?

Chaum's "Mix" paper in CACM (1981? I don't have my refs handy) had a concept where at each step the remailer would encrypt the outgoing non-address part of the message with a DES key found in the anonymous address. The user would remember all the DES keys used in the ARA and un-apply them in reverse order to reconstruct the original message. This would require some special software, I'd think, to remember the DES keys and unapply them (and to construct the anonymous address). (Actually, Chaum didn't specify DES but rather just an unspecified secret key system. If PGP were used for some of this then perhaps IDEA would be a good choice.)

This system sounded pretty complicated, and it still had the problem that by sending multiple messages to the same address a remailer could do some simple traffic analysis and break the ARA. E.g. it would send 5 messages to an ARA today, and discovers that it later gets 5 messages for user X (because it happens to be the last remailer in the ARA chain). Tomorrow, it sends 10 messages to that same ARA, and discovers 10 messages for that same user. The next day it sends 7 messages to the ARA, and discovers 7 messages for that user. Eventually it can deduce that the user and the ARA go together.

To avoid this, Chaum calls these "one-time" ARA's and recommends that mixes not accept messages for the same ARA more than once. I don't think this is a practical suggestion, though, since a one-time ARA is not useful enough.

Hal Finney
74076.1041@compuserve.com

-----BEGIN PGP SIGNATURE-----
Version: 2.1

iQCVAgUBKzcdKKgTA69YIUw3AQFK9wP/a27AgcL4ppMpYIbDfBy6Vw8NTjJzKpsL
hQibQ3XO3wPFURS3Mn51eYLyYRuJPY1/Ve+Y1Kbb6oLiW1u8Btw8CxvB1xe05f32
j0JwzSN1yL8blGMh4JxxXZI0t3SViJ1COt6p+I1SYLjftte/0YX0gc6dweFNUnkZ
5VC4FH2WCbQ=
=+Jx9
-----END PGP SIGNATURE-----

Distribution:
CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM

From: Hal <74076.1041@CompuServe.COM>
Date: Tue, 22 Dec 92 21:53:28 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Signing text messages...

My public key, for those wanting to check the sig on the message below:

-----BEGIN PGP PUBLIC KEY BLOCK-----
mQCNAiqsNkwAAAEEAMKWM52m5EWi0ocK4u1cC2PPyHT6tavk9PC3TB5XBYDegf3d
sldRpnjJj1r+aO08FFO+QLEI9wtBqvf1PPP5iLX7sD2uIVlJH14MPtyVtjm9ZKb8
JMtCW74045BgtHBC9yQ3V7vXNV5jM6dE2ocnH4AI/pBFrGLJPKgTA69YIUw3AAUR
tCZIYWwgRmlubmV5IDw3NDA3Ni4xMDQxQGNvbXB1c2VydmUuY29tPokAlQIFECqu
M1Tidd4O/2f3CwEByrUD/3uoV2y+Fuicrrd2oDawgOw9Ejcx6E+Ty9PVPqKvflLs
0zYyGfeFVSgBbTSDP3X91N3F68nydl9J9VA6QRCGelHM1cZRukCJ0AYbKYfpwUN0
xjEGHsDrd2gT5iWlB3vBZvi+6Ybs4rSq+gyZzVm1/+oRrMen32fz2r0CLgUtHok2
=fF6Z
-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP SIGNED MESSAGE-----

Phil Karn asks about end-of-line conventions for signed text messages.
PGP uses the convention of lines terminated by carriage-return-line-feed.
On Unix systems or other systems which don't use that convention,
it attempts to change the message into this "canonical" text mode
before calculating or checking the signature.

The issue of trailing blanks is more problematical. Some mail gateways
and some mail "user agent" software apparently take liberties with
blanks at the end of lines. The PGP canonical text format does not
include any specification for whether lines could or could not have
blanks at the end. If mailers will leave trailing blanks alone, then
PGP cleartext signed messages will have correct signatures. If some
intervening mailer has added or removed trailing blanks, then the
signatures will be wrong. Presumably something like this has happened
to my signed message on which Edgar found a bad signature. Perhaps
Edgar could try stripping any trailing blanks from his copy of my
message and see if it then signature-checks OK. I'll double-check
that this message is signed with no trailing blanks. Then if you get
a bad signature, I predict that you must have trailing blanks in your
copy of the file. I'd appreciate hearing whether this prediction is
correct.

It would be possible to change PGP's canonical text format
to specify that lines have no blanks at the end. In that case, PGP
would, whenever it computed or checked a signature on a text file,
process the file to make sure that each line ended with a CRLF preceded
by no trailing blanks. I think this would solve a lot of the gateway
problems. But it would be a somewhat more "aggressive" change to what
the user is asking PGP to sign.

The design of PGP's cleartext signature was influenced by PEM, which
also uses a canonical text format for line terminators, but doesn't
deal with trailing spaces, as far as I know.

The real solution, IMO, is to fix those broken mailers that add
or remove spaces. I don't see why this behavior has ever been put
into mail gateways.

Hal Finney
74076.1041@compuserve.com

-----BEGIN PGP SIGNATURE-----
Version: 2.1

iQCVAgUBKzfNHqgTA69YIUw3AQGNdwP/Q51Lvmee1cTb865aInePsRxMTe4qfjeU
DSP8o5hHlBKbII8mCrU/WHZ7upjO3ak4E6wZDyOexsfJFH4FIMnDueihrVVXevlA
FWUQopZIyG4P5Wzofgra8BSjw5WzVZncW2alPQFuB40D9N9VgyopX8IktktVPs4p
qDkHsn9zIpU=
=K/Ui
-----END PGP SIGNATURE-----

Distribution:
CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM

From: Hal <74076.1041@CompuServe.COM>
Date: Tue, 22 Dec 92 21:55:46 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Pax and .fi remailers

Upon more thought, I don't see a really good way to use the PAX
remailer in conjunction with our remailers based on the scripts of
Eric Hughes. The PAX remailer can only be used to send messages to
those who have "registered" with the remailer to receive an anonymous
ID there. So, for PAX to work with our remailers we would have to
register. For example, my remailer at hal@alumni.caltech.edu would
have to register with PAX and receive an anonymous ID, like
"anon.100@pax.tpa.com.au".

Then, to use a two-hop remailer consisting of first PAX and then mine,
you would prepare a message as usual for my remailer:

```
====================
::
Request-Remailing-To: dest
```

```
This is a message for two-hop anonymous remailing.
====================

Perhaps you would encrypt this using my remailer's public key, getting:

====================
::
Encrypted: PGP

-----BEGIN PGP MESSAGE-----
Version: 2.1

hEwCG6rHcT8LtDcBAfwLWYgWXpCoi7TjoeVttBYpk3KPbiYf9L9CCegfYlvj56RA
OFrijYag+jqNlHQXmO52bXL8PaNUowD7a2pFY80WpgAAAGt/RXNzaWkI/b3CkviB
eh/piaUDxgfPd4npcURHtUCEeh8bPpzVaI9qm6xZlxSaJif+CtFqyuaRezj+hcXR
YT9JOl93LAxQJITeYUlPXgkBEvyB4u3HjpCDSS5NETDcqd8rtBspzUvlcmqT1g==
=d356
-----END PGP MESSAGE-----
====================

You would then send this to anon.100@pax.tpa.com.au.  (NOTE: Don't
try this - I haven't yet gotten an anonymous ID at PAX.)  PAX
would forward it to my remailer, unchanged, which would then decrypt
it and send it onward.  Oh, yes, PAX would also strip the .sig,
which is perhaps why you'd want to do this.

But for this to work, I have to publically announce that my remailer,
hal@alumni.caltech.edu, can be reached at PAX "anonymous" address
anon.100@pax.tpa.com.au.  This seems a little strange, as the PAX
address is then no longer anonymous.  I have to tell everybody what
the address is in order for it to be useful.

So, the PAX remailer doesn't really add much anonymity, but it does
excise your .sig.  It's not clear that it's worth it just for that.

On a more positive note, the other remailer, in Finland, is much more
promising for our purposes.  It has a remailing capability similar
to ours.  You could send mail to: hal%alumni.caltech.edu@anon.penet.fi,
and it would forward the mail to hal@alumni.caltech.edu.  This is
similar functionality to a non-encrypting form of the remailers we
are operating, and so it can help confuse things.

Also, this remailer could be used in a chain of our remailers by
using an address of the proper form.  For example, mail sent to
the Rebma remailer, then to Finland, then to the Rosebud remailer, could be
done by putting, at the front of your message:

::
Request-Remailing-To: elee7h5%rosebud.ee.uh.edu@anon.penet.fi

::
Request-Remailing-To: <dest>

Then a blank line, then your message itself.  Mail it to remailer@rebma.mn.org.
I haven't tried this but it should work, theoretically.

Hal
74076.1041@compuserve.com


Distribution:
  CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM
```

```
From: Hal <74076.1041@CompuServe.COM>
Date: Wed, 23 Dec 92 08:42:59 PST
To: CYPHERPUNKS <CYPHERPUNKS@TOAD.COM>
Subject: Re: Pax and .fi remailer


-----BEGIN PGP SIGNED MESSAGE-----

Yanek points out a purpose for a PAX-hidden remailer:

> You don't have to tell anyone that your remailer is behind the anon.100
> address.  You could just (anonymously, of course) announce that a
> remailer is running, and can be reached by sending message to the
> anon.100 address.  This way, no-one but the admins at pax can know
> where the remailer itself lives.  This could be useful in case
```

> remailers are banned.

The problem with this, it seems to me, is that the address of this "secret" remailer is compromised whenever it sends something out. I could just send a "Request-Remailing-To: <me>" message to this PAX anon.100 address, and then look at the return address when the message comes to me from this remailer. So again the anonymity provided by PAX seems to be lost.

Now, one way to avoid this would be for the secret remailer not to send its outgoing mail directly to the requested destination, but rather always to insert one or other remailers into the chain. I think it was Yanek himself (or Dr. Z?) who suggested this earlier. This might work, but as was pointed out, if everyone does this we'll just get into infinite mail loops.

Still, it might be OK if a well-known public remailer were chosen, especially one that was likely to be relativelly immune to pressure. I noted in the discussion of the anon.penet.fi remailer the author made the point that it was running on a machine in his house, one that he owned and used in his independent business. So presumably his machine is not going to be easy to shut down.

(My remailer, OTOH, is running as part of what is basically a guest account on a machine which is to be used just for email and a little telnet/ftp activity. I figure that the remailer performs an email function, speaking broadly, so it's OK under the agreement I signed. But I'm sure that if the admin received some complaints I'd be kicked off. So I can't make any guarantees about how long it will be around.)

(This would be another piece of information that would be useful in the remailer database being constructed by Eric Hollander - some comments on how immune the remailer operator would be to political pressure due to unpopular or illegal messages.)

If this well-known machine guaranteed NOT to do this remail-via-a-remailer outgoing step, then it could be used by less politically secure remailers to protect themselves from pressure. In such a system, Yanek is right that a remailer could run completely anonymously.

Perhaps someone would like to start up a remailer which runs under such a system.

Hal Finney
74076.1041@compuserve.com

-----BEGIN PGP SIGNATURE-----
Version: 2.1

iQCVAgUBKzhqjKgTA69YIUw3AQFNmAQAgioJMosbMCoit2XflfzK/wgIOkG8qBfG
JO3iTWRskVP5Gp43N1bs7W6YhgEXKWdJ/dqNoWrYV2/181zFhXh0xe7lsGifut1b
UQGW6DipYIMlW0TbNjhpiIWwAQChn/3NvTJtcBGL0GY3l4ZjMZFs2qBonc/Y1Boe
jWfWgQbHSXw=
=6y5/
-----END PGP SIGNATURE-----

Distribution:
CYPHERPUNKS >INTERNET:CYPHERPUNKS@TOAD.COM

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Wed, 23 Dec 92 13:46:51 PST
To: cypherpunks@toad.com
Subject: Re: Signing text messages...

Dr. Zaphod writes:

> By including your public key WITH your signed messages, aren't you inviting
> people to intercept it, replace it with they're own public key, and re-sign
> the message? If I didn't have a trusted copy of your public key already I
> wouldn't be able to verify your signature.

I'm not sure what attack you are proposing here. Are you suggesting that someone else could take credit for my (brilliant?) message by removing the PGP signature and substituting one of their own? But digital signatures can't stop other people from doing this.

Or are you suggesting that someone else could create a bogus public key claiming to mine, re-sign the message using that public key, and then get people to think it was from me? Or, worse, they could create a whole new message saying "I am a turkey, signed Hal Finney", sign it with this bogus "Hal Finney" public key, and post it. Then I'd really have egg on my face, right?

But no, I wouldn't, because people would (or should) know not to trust a random public key to be from whom it claims. My posted key is signed by Phil Zimmermann. This doesn't absolutely prove it is from me, but I think it makes it worthwhile to post the key.

Anyway, the real reason I posted the key in this case was so that people could check the cleartext signature to see if it had been mangled by various mail gateways. That was the topic of discussion in the message, so I wanted to make it easy for people to try checking the signature.

Hal
74076.1041@compuserve.com

From: ghsvax!hal@uunet.UU.NET (Hal Finney)
Date: Wed, 30 Dec 92 13:36:33 PST
To: cypherpunks@toad.com
Subject: Return addresses

-----BEGIN PGP SIGNED MESSAGE-----

Eric Messick has an interesting idea in his "postage due" anonymous addresses, where the forwarders would encrypt the message contents as it passed through, and then the receiver would have to pay them to get the message decrypted.

Chaum's idea was that the message contents would be encrypted at each step, as Eric suggests, but Chaum would have the encryption key be part of the anonymous address, created by the same person who made the anonymous address. The idea would be, after decrypting the incoming message, the remailer would see something like:

```
        Anon-To: <next destination>
        Encrypt-With: <some DES or IDEA key>
```

It would then encrypt the message "contents" (but not the "envelope", as Eric points out) using the specified key. When the owner of the anonymous address received the message, he would decrypt it using the chain of "Encrypt-With" keys that he put into the anonymous address.

This does not support Eric's feature of allowing remailers to charge for anonymous addresses, but it does provide more security than the current remailers by changing the message contents at each step.

Hal
74076.1041@compuserve.com

-----BEGIN PGP SIGNATURE-----
Version: 2.1

iQCVAgUBK0IQLqgTA69YIUw3AQHqawQAozCAXrHB1+dksB2fQKeqIoY530340chd
PZlznNGv0wp5gZdIJnFqJ/40scABHjuMc7B7e9QnUglMm1j59b6ZJOGON8kOaYsm
J19vsCOWEWuQhFtMl6oC4hXxPtjZ1BOdm8lr+RQ7KZlpBTe4eusoEMaV5zMMk1TI
vkAT6A4YZ5o=
=DZE1
-----END PGP SIGNATURE-----
